

REMARKS

Claims 1-41 are pending. Claim 29 has been amended. In view of the following remarks, Applicant respectfully solicits allowance of the subject application and furtherance onto issuance.

35 U.S.C. §102(b) and §103(a)

Claims 1-25, 27-39 and 41 are rejected under 35 U.S.C. §102(b) as being anticipated by U.S. Patent No. 5,379,432 to Orton et. al. (hereinafter, "Orton"). Claims 26 and 40 are rejected under 35 U.S.C. §103(a) as being unpatentable over Orton in view of U.S. Pat. No. 5,752,027 to Familiar (hereinafter, "Familiar"). Applicant respectfully traverses the rejections.

Claim 1 is directed to a method of factoring operating system functions that includes defining criteria that governs how functions of an operating system are to be factored into one or more groups. The functions are factored into one or more groups based upon the criteria. Groups of functions are associated with programming objects that have data and methods. The methods correspond to the operating system functions and are effective to provide an object oriented operating system. The programming objects are configured to be instantiated throughout a remote computing system.

The Office asserts InterProcess Communication (IPC) classes of Orton at Col. 26, lines 57-67 and "ReplyAndReceive" at Col. 28, Lines 9-26 for support of the programming object being configured to be instantiated throughout a remote computing system. The Applicant respectfully disagrees. Orton does not disclose, teach or suggest "programming objects being configured to be instantiated throughout a remote computing system" as claimed in Claim 1.

1 Beginning at page 13 of the subject specification, an exemplary distribution
2 is described of an operating system's resources across one process boundary and
3 one machine boundary in a distributed computing system. In the described
4 example, in relation to FIG. 4, resource object 48 is instantiated in-process (i.e.
5 inside the application's process), resource object 50 is instantiated in another
6 process on the same machine (i.e. local), and resource object 52 is instantiated on
7 another machine (i.e. remote). Use of remote and local resources is described in
8 an additional embodiment beginning at page 22 of the subject specification. An
9 operating system is provided with the ability to determine, based on the specified
10 unique identifier, whether it has the resource that is requested. If it does not, the
11 operating system can ascertain the location of the particular resource and retrieve
12 it so that the application can have the requested resource. The location from
13 which the resource is retrieved can be *across process and machine boundaries*.
14 As an example, consider the following. If an application asks for a specific
15 version of a "ReadFile" interface, and the operating system does not support that
16 version, the operating system may know where to go in order to download the
17 code to implement the requested functionality. Software code for the specific
18 requested interface may, for example, be located on a web site to which the
19 operating system has access. The operating system can then simply access the
20 web site, download the code, and provide the resource to the application.

21 Orton is directed to an object-oriented interface for a procedural operating
22 system in the "context of executing the object oriented application 103A on the
23 computer platform 102." *Orton, Col. 8, Lines 21-22*. Orton describes a limited
24 functionality procedural operating system, such as the Mach micro-kernel. *Orton,*
25 *Col. 5, Line 67 to Col. 6., Line 2*. The executable entity in Mach is known as a

1 thread. *Orton, Col. 11, Line 25.* Threads use ports to communicate with each
 2 other. A port is "basically a message queue inside the kernel if [sic] at threads can
 3 add messages to or remove message from, if they have the proper permissions to
 4 do so." *Orton, Col. 15, Lines 39-42.* Ports exist solely in the kernel and can only
 5 be manipulated via port rights. *Orton, Col 15, Lines 48-49.* Thus, Orton describes
 6 that messages are utilized by threads to communicate with each other through the
 7 use of ports.

8 InterProcess Communication (IPC) classes are utilized in Orton to represent
 9 the messages for communication between threads, as shown in the following
 10 excerpt:

11 The IPC classes 410 represent the Mach IPC message
 12 abstraction. Note that all messaging behavior is on the
 13 message classes; the port right classes are basically for
 14 addressing the message. The usage model is preferably as
 15 follows: A TIPCMessageStream is instantiated objects are
 16 streamed into it, and the TIPCMessageStream::Send method
 17 is called with an object representing a destination send-right
 18 passed as an argument. To receive a message, a
 19 TIPCMessageStream is instantiated and its Receive method
 20 called, passing in a receive-right object as an argument.
 21 *Orton, Col. 26, Lines 57-56.*

22 Orton then describes methods for sending/receiving IPC messages, an example of
 23 which was asserted by the Office and is excerpted as follows:

24 ReplyAndReceive (const TPortSendSideHandle&
 25 replyToPort, const TPortReceiveSideHandle& receivePort,
 MIPCMessage& receiveMsg, const TTime&
 timeout=kPositiveInfinity) sends back a reply, blocks and
 receives a new message.

Subclasses' methods for getting/setting port right fields in
 header (Remote and Local Ports: On SEND side, REMOTE
 PORT specifies the destination port, and LOCAL PORT
 specifies the reply port. On RECEIVE side, REMOTE PORT
 specifies the reply port (port to be replied to) and LOCAL
 PORT specifies the port received from. The way the port was

(or is to be) transmitted is returned in theDisposition. It can have values: MACH.sub.-- MSG.sub.-- TYPE.sub.-- (MOVE.sub.-- RECEIVE, MOVE.sub.-- SEND, MOVE.sub.-- SEND.sub.-- ONCE, COPY.sub.-- SEND, MAKE.sub.-- SEND, MAKE.sub.-- SEND.sub.-- ONCE}). GetRemotePort: pass in the remote port right, and specify the disposition. *Orton, Col. 28, Lines 9-26.*

It is respectfully submitted that the Office has misinterpreted the use of "interprocess communication" as indicating remote communication. This is not the case. As shown in the above excerpts, Orton describes communication between threads in the "context of executing the object oriented application 103A on the computer platform 102." *Orton, Col. 8, Lines 21-22.* This is further supported by the Summary of the Invention in Orton, which is excerpted as follows:

The present invention is directed to a system and method of enabling an object-oriented application to access in an object-oriented manner a procedural operating system having a native procedure interface. *The system includes a computer and a memory component in the computer. Orton, Col. 3, Lines 50-55 (emphasis added).*

Therefore, the system described in Orton is implemented in a single computer. Indeed, nowhere in Orton is remote communication or a machine boundary even discussed. Rather, Orton describes communication between an application and an operating system that are executed on a computer platform, such as "an International Business Machines (IBM) computer or an IBM-compatible computer [or an] Apple computer". *Orton, Col. 6, Lines 6-10.* Although Orton describes "interprocess communication", Orton does not disclose, teach or suggest communication in *a remote computing system or across machine boundaries.* Therefore, *Orton* does not disclose, teach or suggest "programming objects being

1 configured to be instantiated throughout a remote computing system” as claimed
2 in Claim 1. According, for at least these reasons, this claim is allowable.

3 **Claims 2-13** depend either directly or indirectly from claim 1 and are
4 allowable as depending from an allowable base claim. These claims are also
5 allowable for their own recited features which, in combination with those recited
6 in claim 1, are neither shown nor suggested in the references of record, either
7 singly or in combination with one another. For example, claim 7 recites
8 “instantiating a plurality of programming objects across a machine boundary”.
9 Orton does not disclose, teach or suggest the presence of a machine boundary nor
10 even mention more than one computer platform. Therefore, for at least this
11 reason, claim 7 is allowable.

12 **Claim 14** recites a method of factoring operating system functions. A
13 plurality of operating system functions that are used in connection with operating
14 system resources into first groups based upon first criteria. The first groups are
15 also factored into individual sub-groups based upon second criteria. Each sub-
16 group is assigned to its own programming object interface. A programming object
17 interface represents a particular object’s implementation of its collective methods
18 that are effective to provide an object-oriented operating system. Individual
19 objects having associated programming object interfaces are configured to be
20 instantiated throughout a remote computing system.

21 As previously described, Orton neither discloses nor suggests a method that
22 factors operating system functions where “individual objects having associated
23 programming object interfaces are configured to be instantiated throughout a
24 remote computing system” as claimed in claim 14. Accordingly, for at least this
25 reason, claim 14 is allowable.

1 **Claims 15-23** depend either directly or indirectly from claim 14 and are
2 allowable as depending from an allowable base claim. These claims are also
3 allowable for their own recited features which, in combination with those recited
4 in claim 14, are neither shown nor suggested in the references of record, either
5 singly or in combination with one another.

6 **Claim 24** recites a method of factoring operating system functions which
7 includes factoring a plurality of operating system functions into interface groups
8 based upon the resources with which a function is associated. The interface
9 groups are factored into interface sub-groups based upon each function's use of a
10 handle that represents a resource. The interface sub-groups are organized so that
11 at least one of the interface sub-groups inherits from at least one other of the
12 interface sub-groups. Individual interface sub-groups are associated with
13 individual programming objects that can be instantiated throughout a remote
14 computing system.

15 Again, as previously described, Orton neither discloses nor suggests a
16 method that factors operating system functions where "individual objects sub-
17 groups being associated with individual programming objects that can be
18 instantiated throughout a remote computing system" as claimed in claim 24.
19 Accordingly, for at least this reason, claim 24 is allowable.

20 **Claims 25 and 27-28** depend from claim 24 and are allowable as
21 depending from an allowable base claim. These claims are also allowable for their
22 own recited features which, in combination with those recited in claim 24, are
23 neither shown nor suggested in the references of record, either singly or in
24 combination with one another.

1 **Claim 26** is rejected over Orton in view of Familiar. Familiar does not
2 cure the defects of Orton, namely "individual objects sub-groups being associated
3 with individual programming objects that can be instantiated throughout a remote
4 computing system" as claimed in claim 24. Therefore, given the allowability of
5 claim 24, claim 26 is allowable over both Orton and Familiar, alone and in
6 combination.

7 **Claim 29** has been amended and, as amended, recites an operating system
8 application program interface embodied on a computer-readable medium
9 comprising a plurality of object interfaces. Each object interface is recited to be
10 associated with an object that includes one or more methods that are associated
11 with and can call functions of an operating system that does not comprise the
12 object interfaces. At least one object is configured *to be remotely instantiated*.
13 Nowhere does Orton disclose or even suggest any such subject matter.
14 Accordingly, this claim is allowable.

15 **Claims 30-35** depend either directly or indirectly from claim 29 and are
16 allowable as depending from an allowable base claim. These claims are also
17 allowable for their own recited features which, in combination with those recited
18 in claim 29, are neither shown nor suggested in the references of record, either
19 singly or in combination with one another.

20 **Claim 36** recites an operating system that includes a plurality of
21 programming objects having interfaces. The programming objects represent
22 operating system resources, and the interfaces define methods that are organized in
23 accordance with whether they create an operating system resource or not. The
24 programming objects are configured to be called either directly or indirectly by an
25 application. The methods are configured to call operating system functions

1 responsive to being called directly or indirectly by an application. The
2 programming objects are configured to be instantiated throughout a remote
3 computing system.

4 As previously described, Orton neither discloses nor suggests
5 "programming objects being configured to be instantiated throughout a remote
6 computing system" as claimed in claim 36. Accordingly, for at least this reason,
7 claim 36 is allowable.

8 Claims 37-39 depend from claim 36 and are allowable as depending from
9 an allowable base claim. These claims are also allowable for their own recited
10 features which, in combination with those recited in claim 36, are neither shown
11 nor suggested in the references of record, either singly or in combination with one
12 another.

13 Claim 40 is rejected over Orton in view of Familiar. Familiar does not
14 cure the defects of Orton, namely "programming objects being configured to be
15 instantiated throughout a remote computing system" as claimed in claim 36.
16 Therefore, given the allowability of claim 36, claim 40 is allowable over both
17 Orton and Familiar, alone and in combination.

18 Claim 41 recites a method of converting an operating system from a non-
19 object-oriented format to an object oriented format. The operating system
20 includes a plurality of operating system functions that are callable to create or use
21 operating system resources. A plurality of programming object interfaces are
22 defined that define methods that correspond to the operating system functions.
23 Programming objects that support the interfaces are callable either directly by an
24 application that makes object-oriented calls, or indirectly by an application that
25 makes function calls. The programming objects being configured to be

1 instantiated throughout a remote computing system. A programming object
2 interface is called either directly via an object-oriented call, or indirectly via an
3 indirection that transforms a function call into an object-oriented call. Responsive
4 to the calling, an operating system function is called with a method of the
5 programming object that supports said programming object interface. Orton
6 neither discloses nor suggests "programming objects being configured to be
7 instantiated throughout a remote computing system" as claimed in claim 41.
8 Accordingly, for at least this reason, claim 36 is allowable.

9 For at least these reasons, claims 1-25, 27-39 and 41 are allowable over
10 Orton. Applicant respectfully requests that the §102 rejection of claims 1-25, 27-
11 39 and 41 be withdrawn.

12 Additionally, Claims 26 and 40 are allowable over Orton in view of
13 Familiar. Applicant respectfully requests that the §103 rejection of claims 26 and
14 40 be withdrawn.

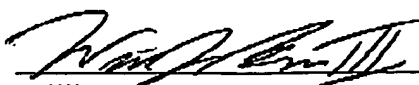
15
16 **Conclusion**

17 All pending claims 1-41 are in condition for allowance. Applicant
18 respectfully requests reconsideration and prompt issuance of the subject
19 application. If any issues remain that prevent issuance of this application, the
20 Examiner is urged to contact the undersigned attorney before issuing a subsequent
21 Action.

22
23 Respectfully Submitted,
24
25

Dated: December 11, 2003

By:


 William J. Breen, III
 Reg. No. 45,313
 (509) 324-9256 x249

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25